



14 | 百科应用系统设计：机房被火烧了系统还能访问吗？

2022-03-21 李智慧

《李智慧·高并发架构实战课》

[课程介绍 >](#)



讲述：李智慧

时长 12:28 大小 11.42M



你好，我是李智慧。

百科知识应用网站是互联网应用中一个重要的类别。很多人上网是为了获取知识，而互联网上的信息良莠并存，相对说来，百科知识应用网站能为普通人提供较为可信的信息。因此，百科知识网站虽然功能单一、设计简单，但是依然撑起了互联网的一片天空：维基百科是全球访问量 TOP10 的网站，百度百科是百度的核心产品之一。

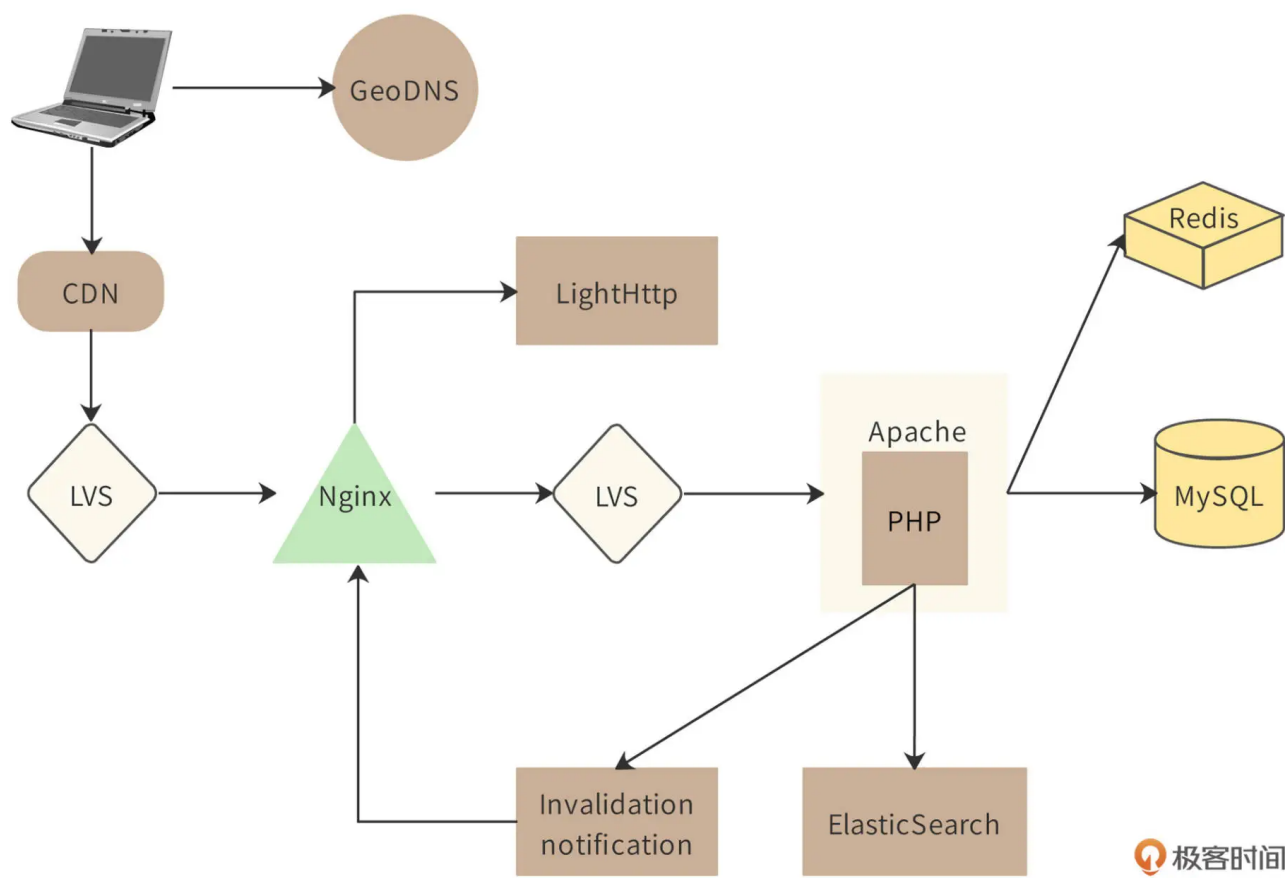
我们准备开发一个供全球用户使用的百科知识应用系统，系统名称为 “Wepedia”。



Wepedia 的功能比较简单，只有编辑词条和搜索查看词条这两个核心功能。但是 Wepedia 的设计目标是支撑每日 10 亿次以上的访问压力。因此设计目标主要是简单、高效地支持高并发访问，以及面对全球用户时保证 7×24 小时高可用。

概要设计

Wepedia 的整体架构，也就是简化的部署模型如图。



在梳理 Wepedia 整体逻辑之前，先说明下架构图中核心组件的作用。

组件	作用
GeoDNS	基于地理位置的域名解析服务。 可将域名www.wepedia.org解析到离用户最近的那个数据中心。
CDN	内容分发网络，部署在网络运营商机房的一种缓存服务器。 因为离用户距离近，所以CDN可以更快速地响应用户请求，加快访问速度。同时，CDN还能极大降低数据中心的访问压力。
LVS	基于 Linux 的开源负载均衡服务器。 将高并发的用户请求均匀分发到多个服务器，使每个服务器上处理的用户请求都不超过自己的负载处理能力。
Nginx	反向代理服务器，主要提供缓存功能。 用户访问某个词条时，Nginx先查找自己服务器上是否有缓存该词条内容，如果有，就直接对应内容；如果没有，Nginx就会访问后面的Apache应用服务器，获取该词条的内容。Nginx得到Apache的返回内容后，一方面将该词条内容响应给用户，一方面缓存在自己的服务器上，以备其他用户访问。
Lighttpd	开源的HTTP服务器。 较Apache 服务器更轻量、更快速。Wepedia使用 Lighttpd 作为HTTP图片服务器，图片上传、存储、下载都通过该服务器完成。
Redis	高性能的开源分布式缓存系统。 Wepedia使用Redis作为对象缓存，缓存词条内容对象。
Elasticsearch	分布式的搜索引擎服务器。 通过索引的方式进行内容查找，搜索速度极快。

用户在 Web 端查看一个百科词条的时候，首先通过 GeoDNS 进行域名解析，得到离用户最近的数据中心所属的 CDN 服务器的 IP 地址。用户浏览器根据这个 IP 地址访问 CDN 服务器，如果 CDN 服务器上缓存有用户访问的词条内容，就直接返回给用户；如果没有，CDN 会访问和自己在同一个区域的 Wepedia 的数据中心服务器。

准确地说，CDN 访问的是 Wepedia 数据中心负载均衡服务器 LVS 的 IP 地址。请求到达 LVS 后，LVS 会将该请求分发到某个 Nginx 服务器上。Nginx 收到请求后，也查找自己服务器上是否有对应的词条内容，如果没有，就将请求发送给第二级 LVS 负载均衡服务器。

接着，第二级 LVS 将请求分发给某个 Apache 服务器，Apache 会调用 PHP 程序处理该请求。PHP 程序访问 Redis 服务器集群，确认是否有该词条的对象。如果有，就将该对象封装成 HTML 响应内容，返回给用户；如果没有，就访问 MySQL 数据库来查找该词条的数据内容。PHP 程序一方面会将 MySQL 返回的数据构造成对象，然后封装成 HTML 返回用户，一方面会将该对象缓存到 Redis。

如果用户的 HTTP 请求是一个图片，那么 Nginx 则会访问 LightHttp 服务器，获取图片内容。

因为 Nginx 缓存着词条内容，那么当词条编辑者修改了词条内容时，Nginx 缓存的词条内容就会成为脏数据。解决这个问题通常有两种方案，一种是设置失效时间，到了失效时间，缓存内容自动失效，Nginx 重新从 Apache 获取最新的内容。但是这种方案并不适合 Wepedia 的场景，因为词条内容不会经常被编辑，频繁失效没有意义，只是增加了系统负载压力；而且，在失效时间到期前，依然有脏数据的问题。

Wepedia 为了解决 Nginx 缓存失效的问题，采用了另一种解决方案：失效通知。词条编辑者修改词条后，Invalidation notification 模块就会通知所有 Nginx 服务器，该词条内容失效，进而从缓存中删除它。这样，当用户访问的时候，就不会得到脏数据了。

多数据中心架构

Wepedia 在全球部署多个数据中心，可以就近为用户提供服务。因为即使是最快的光纤网络，从地球一端访问另一端的数据中心，在通信链路上的延迟就需要近 150ms。

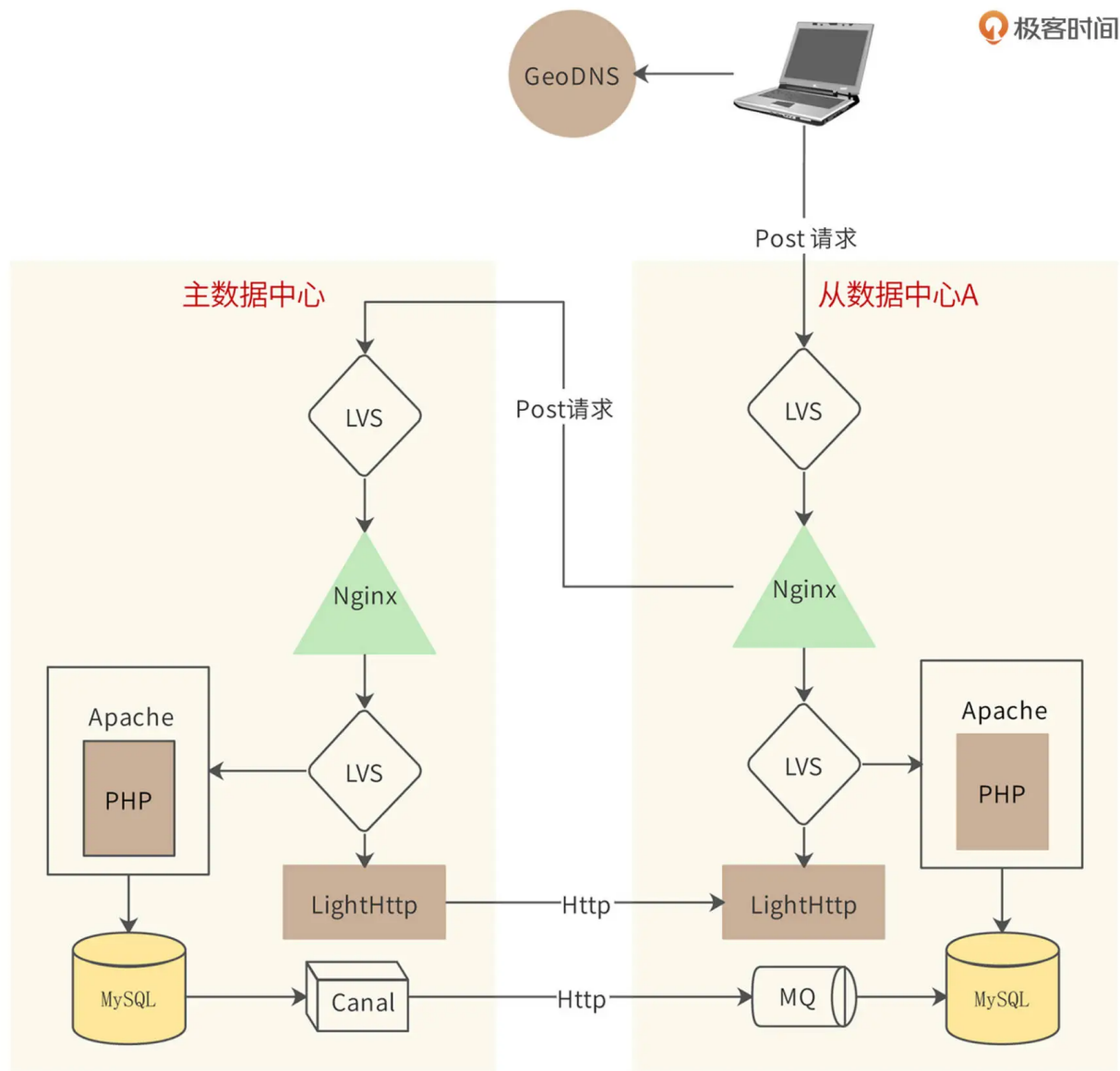
$$(\text{地球周长} 4\text{万} KM \div 2) \div \text{光速} 30\text{万} KM/s \times \text{请求响应} 2\text{次通信} \approx 133ms$$

150ms 是一个人类能够明显感知的卡顿时间。再加上服务器的处理时间，用户的响应等待时间可能会超过 1 秒钟，而页面加载时间超过 1 秒钟，用户就会明显不耐烦。多数据中心架构可以通过 GeoDNS 为用户选择最近的数据中心服务器，减少网络通信延迟，提升用户体验。

另一方面，多数据中心还具有容灾备份功能，如果因为天灾或者人祸导致某个数据中心机房不可用，那么用户还可以访问其他数据中心，保证 Wepedia 是可用的。

但是多数据中心需要解决**数据一致性**的问题：如果词条编辑者修改词条内容，只记录在距离自己最近的数据中心，那么这份数据就会和其他数据中心的**不一致**。所以，Wepedia 需要在多个数据中心之间进行数据同步，用户不管访问哪个数据中心，看到的词条内容都应该是一样的。

Wepedia 的多数据中心架构如图。



Wepedia 的多数据中心架构为一主多从架构，即一个主数据中心，多个从数据中心。如果用户请求是 Get 请求（读请求），那么请求就会在该数据中心处理。如果请求是 Post 请求（写请求），那么请求到达 Nginx 的时候，Nginx 会判断自己是否为主数据中心，如果是，就直接在该数据中心处理请求；如果不是，Nginx 会将该 Post 请求转发给主数据中心。

通过这种方式，主数据中心根据 Post 请求更新数据库后，再通过 Canal 组件将更新同步给其他所有从数据中心的 MySQL，从而使所有数据中心的数据保持一致。同样，LightHttp 中的图片数据也进行同步，开发 LightHttp 插件，将收到的图片，发送给所有从数据中心。

数据中心之间采用类似 ZooKeeper 的选主策略进行通信，如果主数据中心不可用，其他数据中心会重新选举一个主数据中心。而如果某个从数据中心失火了，用户请求域名解析到其他数据中心即可。

这种多数据中心架构虽然使词条编辑操作的时间变长，但是由于 Wepedia 的绝大多数请求都是 Get 请求（Get 与 Post 请求比超过 1000 : 1），因此对系统的整体影响并不很大。同时用一种简单、廉价的方式实现多数据中心的数据一致性，开发和运维成本都比较低。

详细设计

作为一个百科服务类网站，Wepedia 主要面临的挑战是：应对来自全球各地的巨量并发的词条查询请求。因此详细设计重点关注 Wepedia 的性能优化。

前端性能优化

前端是指应用服务器（也就是 PHP 服务器）之前的部分，包括 DNS 服务、CDN 服务、反向代理服务、静态资源服务等。对 Wepedia 而言，80% 以上的用户请求可以通过前端服务返回，请求根本不会到达应用服务器，这也就使得网站最复杂、最有挑战的 PHP 应用服务端和存储端压力骤减。

Wepedia 前端架构的核心是反向代理服务器 Nginx 集群，大约需要部署数十台服务器。请求通过 LVS 负载均衡地分发到每台 Nginx 服务器，热点词条被缓存在这里，大量请求可直接返回响应，减轻应用负载压力。而 Nginx 缓存不能命中的请求，会再通过 LVS 发送到 Apache 应用服务器集群。

在反向代理 Nginx 之前，是 CDN 服务，它对于 Wepedia 性能优化功不可没。因为用户查询的词条大部分集中在比重很小的热点词条上，这些词条内容页面缓存在 CDN 服务器上，而 CDN 服务器又部署在离用户浏览器最近的地方，用户请求直接从 CDN 返回，响应

速度非常快，这些请求甚至根本不会到达 Wepedia 数据中心的 Nginx 服务器，服务器压力减小，节省的资源可以更快地处理其他未被 CDN 缓存的请求。

Wepedia CDN 缓存的几条准则：

1. 内容页面不包含动态信息，以免页面内容缓存很快失效或者包含过时信息。
2. 每个内容页面有唯一的 REST 风格的 URL，以便 CDN 快速查找并避免重复缓存。
3. 在 HTML 响应头写入缓存控制信息，通过应用控制内容是否缓存及缓存有效期等。

服务端性能优化

服务端主要是 PHP 服务器，这里是业务逻辑的核心部分，运行的模块都比较复杂笨重，需要消耗较多的资源，Wepedia 需要将最好的服务器部署在这里（和数据库配置一样的服务器），从硬件上改善性能。

除了硬件改善，Wepedia 还需要使用其他开源组件对应用层进行优化：

1. 使用 APC，这是一个 PHP 字节码缓存模块，可以加速代码执行，减少资源消耗。
2. 使用 Tex 进行文本格式化，特别是将科学公式内容转换成图片格式。
3. 替换 PHP 的字符串查找函数 strstr()，使用更优化的算法重构。

存储端性能优化

包括缓存、存储、数据库等被应用服务器依赖的服务都可以归类为存储端服务。存储端服务通常是一些有状态的服务，即需要进行数据存储。这些服务大多建立在网络通信和磁盘操作基础上，是性能的瓶颈，也是性能优化的关键环节。

存储端优化最主要的手段是使用缓存，将热点数据缓存在分布式缓存系统的内存中，加速应用服务器的数据读操作速度，减轻存储和数据库服务器的负载。

Wepedia 的缓存使用策略如下：

1. 热点特别集中的数据直接缓存到应用服务器的本地内存中，因为要占用应用服务器的内存且每台服务器都需要重复缓存这些数据，因此这些数据量很小，但是读取频率极高。

2. 缓存数据的内容尽量是应用服务器可以直接使用的格式，比如 HTML 格式，以减少应用服务器从缓存中获取数据后解析构造数据的代价。
3. 使用缓存服务器存储 session 对象。

作为存储核心数据资产的 MySQL 数据库，需要做如下优化：

1. 使用较大的服务器内存。在 Wepedia 应用场景中，增加内存比增加其他资源更能改善 MySQL 性能。
2. 使用 RAID5 磁盘阵列以加速磁盘访问。
3. 使用 MySQL 主主复制及主从复制，保证数据库写入高可用，并将读负载分散在多台服务器。

小结

高可用架构中的各种策略，基本上都是针对一个数据中心内的系统架构、针对服务器级别的软硬件故障而进行设计的。但如果整个数据中心都不可用，比如数据中心所在城市遭遇了地震，机房遭遇了火灾或者停电，不管我们架构的设计多么的高可用，应用依然是不可用的。

为了解决这个问题，同时也为了提高系统的处理能力、改善用户体验，很多大型互联网应用都采用了异地多活的多机房架构策略，也就是说将数据中心分布在多个不同地点的机房里，这些机房都可以对外提供服务。用户可以连接任何一个机房进行访问，这样每个机房都可以提供完整的系统服务，即使某一个机房不可使用，系统也不会宕机，依然保持可用。

思考题

词条编辑者修改词条的时候，可能会同时修改（新增）词条文本和图片。而数据从主数据中心同步到多个从数据中心的时候，数据库同步可能和图片同步时间不一致，导致用户查看词条的时候，图片无法加载或者图片和文本内容不一致。

如何解决这个问题？

附 1：阿里巴巴在十几年前，也遇到数据和图片同步不一致的问题，后来解决这个问题的开发工程师晋升为阿里集团副总裁，欢迎有志于成为副总裁的同学思考下这个问题。

附 2：阿里当年遇到并解决这个问题的系统：<https://github.com/alibaba/otter>

附 3：阿里当年解决这个问题的工程师访谈：<https://www.infoq.cn/article/pl-alibaba>

欢迎在评论区分享你的思考，我们共同进步。

分享给需要的人，Ta订阅超级会员，你最高得 50 元

Ta单独购买本课程，你将得 20 元

 生成海报并分享

 赞 0  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 13 | 微博系统设计：怎么应对热点事件的突发访问压力？

下一篇 15 | 限流器设计：如何避免超预期的高并发压力压垮系统？

精选留言 (7)

 写留言



Lance-Yanh

2022-03-27

从数据中心重放DDL的时候同步或异步检查图片是否同步完成，进一步确定是否延时重放DDL或主动同步完图片再重放。这样可以吗？老师



javaadu 

2022-03-25

思考题，图片和文字属于某一个词条，那就将这个词条作为数据同步的最小粒度，利用词条信息将文本和图片绑定在一起

**丘振标**

2022-03-21

感觉思考题本质上是要解决分布式环境下，数据同步的顺序性问题，对于词条的文本和图片，在主数据中心提交的时候，一般会按照先保存图片然后再更新文本的顺序(否则，在主数据中心这里，就算不存在同步的因素，也会有问题)，因此，这里应该有个统一的同步系统按顺序将变更逐一提交到从数据中心，有点类似zookeeper的master节点逐一同步数据到其他follower的原理，只要保证了同步系统同步数据的顺序性，思考题中说的问题应该就...
展开 ∨

作者回复: 你括号中提到的问题还真是问题，在主数据中心，写入图片和写入数据的服务器都不是同一组服务器，也就是说，主数据中心自己都不知道自己的图片和数据是不是都写入完成了，如何做顺序同步呢？

而且ZooKeeper是不能并发写的，而Wepedia是高并发写需求的，逐一顺序同步这个恐怕也不能接受。

**启程**

2022-03-21

主数据中心有几个？是多个吗？主主之间怎么写数据，是谁来负责选择哪个主数据中心写入，同时主主之间数据怎样同步，老师能详细解答一下吗

作者回复: 原文：Wepedia 的多数据中心架构为一主多从架构，即一个主数据中心，多个从数据中心。数据中心之间采用类似 ZooKeeper 的选主策略进行通信，如果主数据中心不可用，其他数据中心会重新选举一个主数据中心。



2022-03-21

我觉得核心处理原理的话，raft协议。etcd和zookeeper使用的协议。保障数据一致性的解决方案。所有的节点之间有一个选主的流程。非主节点只允许读操作。并且，每一条数据都会有一个版本号，主节点修改或者创建数据时，会同步通知非主节点，当所有的非主节点都告知主节点数据更新成功时，才会返回这条最新的数据。但是，因为其数据强一致性的保障，牺牲了很多性能。这种方式是否合理也是一个问题。

展开 ∨

作者回复: 问题描述虽然用“数据一致”，其实想表达的意思是数据库数据和图片数据一致，真正的意思其实是“数据完整”，用分布式数据一致性的方案解决我感觉可能不是合适的。

共 2 条评论 >



奔浪

2022-03-21

原因:由于图片资源同步数据时耗时较长导致同步数据时有可能图文不符，或者图片资源干脆加载不到，个人愚见: 方案一是否可以先进行图片同步文本类型先等等 等待图片同步完成后再写入数据库，因为文本写入与同步较快，像这种软件一般应该是富文本编辑器吧？在同步文本数据时内有img标签资源地址，这样在同步过程中，要么访问到是老资源，要么就是新资源。 方案二是否将文本文件与图片资源整合成一张图片资源，这样就不存在具...
展开 v

作者回复: 不错，有两个小问题

方案1：如何判断图片同步完成呢？图片和数据同步是各自独立的，图片自己并不知道自己被哪个数据引用，而图片和数据也是在高并发同时有很多在更新的。

方案2：虽然不知道一张图片如何整合文本文件和图片，但是我觉得这个方案挺靠谱的~~~



peter

2022-03-21

请教老师几个问题啊： Q1：CDN、Nginx、Redis缓存能力比较？ 本文架构中，CDN、Nginx、Redis的作用主要是缓存。那么，这三种谁的单机缓存能力最强？ 我的猜测是：1 单机缓存能力：Redis > CDN > Nginx。 2 Redis和CDN靠机器数量提高缓存能力。 Q2：Invalidation Notificatin，是Apache应用服务器的一个模块？还是一个独立的服务器？ Q3：多数数据中心同步时，未同步完时，通过从数据中心读，会读到脏数据，怎么处理？ Q4：MQ...
展开 v

作者回复: 1 单机缓存能力强不强，就是看缓存程序的实现算法优不优。Redis和Nginx是程序，CDN是一个技术方案；Redis只用作缓存，Nginx用途可多了，三者不好比较啊。

2 是系统中的一个子系统，需要自己开发，部署独立的服务器。

3 数据同步延迟导致的数据中心之间的数据不一致，在百科应用场景下，业务上不认为这是脏数据。

4 是的，画图简化了，需要消费者。

5 敢这么说话的都是大佬~~~ wepedia这么设计其实是借鉴了wikipedia



